Benchmarking API Costs of Network Sampling Strategies

Michele Coscia*†, Luca Rossi†
*Harvard University
michele_coscia@hks.harvard.edu
†IT University of Copenhagen
{mcos,lucr}@itu.dk

***Abstract*—Online social media contain valuable quantitative and qualitative data, necessary to advance complex social systems studies. However, these data vaults are often behind a wall: the owners of the media sites dictate what, when, and how much data can be collected via a mandatory interface (called Application Program Interface: API). To work with such restrictions, network scientists have designed sampling methods, which do not require a full crawl of the data to obtain a representative picture of the underlying social network. However, such sampling methods are usually evaluated only on one dimension: what strategy allows for the extraction of a sample whose statistical properties are closest to the original network? In this paper we go beyond this view, by creating a benchmark that tests the performance of a method in a multifaceted way. When evaluating a network sampling algorithm, we take into account the API policies and the budget a researcher has to explore the network. By doing so, we show that some methods which are considered to perform poorly actually can perform well with tighter budgets, or with different API policies. Our results show that the decision of which sampling algorithm to use is not monodimensional. It is not enough to ask which method returns the most accurate sample, one has also to consider through which API constraints it has to go, and how much it can spend on the crawl.**

***Keywords*-Network Sampling; Online Social Network; Social Media;**

I. Introduction

Online social media represent an unprecedented opportunity for studying complex social systems and phenomena. They provide a wealth of data in machine-readable format that was previously hard to gather, prone to noise, and usually difficult to analyze automatically. For instance, reconstructing a network of friendship in the past required pen-and-paper questionnaires [33], which limited the size of the studied social networks to a few dozen nodes. Today, on Facebook, we can investigate social networks with hundreds of millions of nodes[1].

However, the vast majority of social media platforms recognize data as one of their primary source of revenue, and therefore put restrictions on the amount of data you can extract from them per unit of time. Their terms of service usually state that researchers have to use a custom interface (API). The owners of the social media dictate how their API works, establishing what and how much data each request can obtain, and how and when such requests can be made. Under such limitations, to fully crawl Twitter's social network requires a time and labor investment beyond the capability of most – if not all – research groups. Twitter has 330 million monthly active users[2] and a restriction of one user crawl per minute [5]. A non-parallel crawl of the follower network would then take more than 600 years.

Even before the advent of large scale social networks, network scientists were addressing this problem through so-called sampling methods. However, to the best of our knowledge, when a new network sampling method is proposed, the only criterion to evaluate its performance is testing whether the samples it returns more closely represent the original properties of the full network. Usually these properties include the degree distribution, the clustering/modularity of the network, among many others. In this paper, we propose that there are some unexplored trade-offs involved and we consider their investigation to be the main contribution of this paper. For instance, consider crawling cost: perhaps some methods do not represent the original network well on a large budget, but outperform the alternatives on a tighter one. Some API policies might make previously underperforming sampling methods better than the alternatives.

We focus on topological sampling methods, which explore the network one edge at a time. There are other methods such as node- and edge-samples, which select random nodes or edges and then induce the network by collecting all nodes and edges directly connected to their sample. However, such methods cannot be applied to the typical social media API, and therefore they are outside the scope of this paper. Moreover, they usually create a biased sample, by overestimating the exponent of the degree distribution [26].

In this paper we create a benchmark for testing the trade-off between the cost of a network sampling strategy and the representativeness of the sample it creates. We allow researchers to specify an arbitrary API cost policy, and to test which sampling method works best given this policy and their budget constraints.

As motivation, we simulate the costs of many popular topological network sampling algorithms, using different API policies, different network topologies, and different attribute

[1]https://en.wikipedia.org/wiki/Facebook\#User_growth, date of access: April 16th, 2018

[2]https://www.statista.com/statistics/282087/number-of-monthly-active-twitter-users/, date of access: May 11th, 2018

distributions. By showing how different algorithms perform in these scenarios, we show that a researcher with the task of extracting a social network from a social media has to consider multiple factors when selecting a sampling strategy. There is not a single best method to obtain an unbiased sample in every possible scenario. API policy costs and budget constraints also play a role in such a decision.

From our results, it appears clear that network sampling is similar to community discovery in that there is no free lunch [37]: we cannot point to a single method outperforming all other methods in any arbitrary scenario – although we leave the proof of the theorem to future work. Moreover, we also show that there is a complex relationship between the total throughput of an API system, and the ability of sampling methods to faithfully represent the underlying network structure. Higher-throughput APIs result in less representative samples, given specific conditions.

We consider this paper a timely addition to the network sampling literature, given that we expect the online social media gathering task to become significantly harder in the future. In the wake of data handling misbehavior by private companies, some platforms have already increased their API restrictions[3]. We release our benchmark system as an open source library for public use[4].

II. Related Work

This paper is about evaluating topological network sampling algorithms. Network sampling is a crucial problem at the heart of social network analysis. Almost all papers and subtopics in social network analysis face the problem of extracting a representative sample from social media. Examples of applications range from detecting socially prominent users [38], the effect of missing data [42], knowledge base building [48], and damage detection [16].

There is a vast literature on evaluating sampling algorithms. The majority of the papers in it are interested in determining which strategy yields the most representative sample of a network [14], [20], [21], [18]. Sometimes, constraints are added. For instance, some works focus on the scenario of hidden populations isolated from the rest of society [23], which are notoriously hard to sample from (e.g. drug users [13]). In other cases, the underlying graph is not static, but it can significantly change during the sampling process [39]. In yet another case, an application-oriented strategy is evaluated – for instance in aiding A/B testing [10].

In none of these cases, however, different API policies of different platforms are taken into account. Also, researchers usually fix a budget for all methods and the tests they run, and rarely explore what changing that budget implies for different costs as defined by the API system. APIs and their cost policies have been considered in discussions on how to complete a partial network representation [43], [44], but not in discussions of how to extract a sample from a network reflecting its global properties.

[3]https://techcrunch.com/2018/04/02/instagram-api-limit/, date of access: April 16th, 2018.

[4]http://www.michelecoscia.com/?page_id=1390

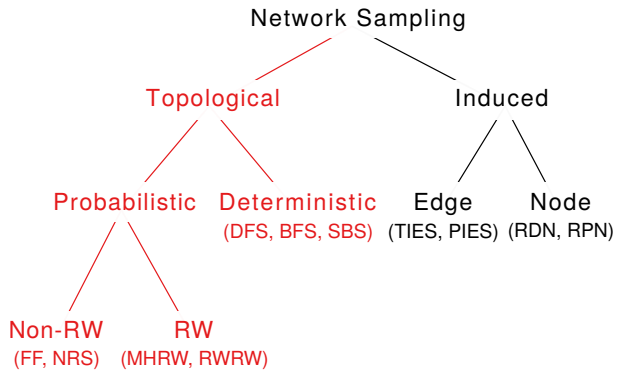


Fig. 1: Our classification of network sampling strategies. We highlight in red the branch on which we focus in this paper.

Figure 1 depicts the rough organization of network sampling strategies we follow in this paper. The main families of sampling methods are topological and induced.

In induced sampling, one determines a set of nodes (node induced) or a set of edges (edge induced) and then generates a sample by collecting all nodes/edges adjacent to the sample. Popular edge sample methods are Partially and Totally Induced Edge Samples (TIES and PIES [7], [8]), while node sample methods are either fully uniform, or degree (RDN) or PageRank weighted samples (RPN) – see [27] for a discussion of these methods. We choose not to focus on these methods, because usually social media APIs do not allow for an easy application of such strategies.

In topological sampling, one chooses a random starting point and then explores the network with a walk along its edges. The walk can be deterministic – as in classical Depth or Breadth First Search or Snowball sampling – or probabilistic – random walks [28], [45], [24], [41], [40] or probabilistic modification of the deterministic approaches such as forest fire [27] or Neighbor Reservoir Sampling [29], [9]. Since these methods are the main focus of the paper, we discuss them more in depth in Section IV.

In conclusion, no matter how accurately we can make a network sampling method and how realistically we can benchmark and evaluate it, API-based sampling is affected by fundamental issues. Works in the past have shown that one has to be careful when working with data sources that potentially yield non-representative samples of the phenomenon at large [35], [34].

III. Problem Definition

Here, a topological network sampling approach is seen as a function f taking four inputs: $G = (V, E)$, the original graph – with V being the set of nodes and $E \subseteq V \times V$ the set of edges –; v, the seed node from which to start the exploration of G; b, the available budget; and P, the API cost policies. The constraint on f is that it has to start exploring G from v, and it can only follow edges in E. The result of f is $G' = (V', E')$ – with $V' \subseteq V$ and $E' \subseteq E$ –, e.g. a proper subgraph of G.

We assume to have a quality function $q_G(f)$ for any arbitrary G that, given a function f, returns the similarity between the topological properties of G and G', the result

of f. We can then state the general problem definition that all topological network sampling algorithms aim to solve as follows:

Definition 1 (Topological Network Sampling): Given a graph G, a seed node v, a budget b, a set of cost policies P, and a quality function q, design a function f which explores G from v and returns $G' = (V', E')$, such that $P(f, G) \leq b$ and $f = \arg\max_f q_G(f)$.

Our problem definition is how to create a benchmark system B which, given a collection of fs, is able to evaluate their q_G performances for varying budgets b in presence of constraints P. Traditionally, when introducing a new sampling method, it is evaluated using only one of the many possible q_G. However, in practice, we have an ensemble of functions q_G, one for each of G's properties. Moreover, P and b take values in a large space and, traditionally, the impact of P has been under-explored. This makes optimizing f difficult, and creates the necessity for the benchmark system B.

In the following subsections, we explore more closely the internal structure of P. P includes cost policies regulating the collection of edges (Section III-A) and dictating how to space out queries to the API system in time (Section III-B). The first subsection defines how to calculate the number of API calls needed to collect the edges. The second subsection shows how to convert this number of calls into the crawl cost – as we consider it in this paper –: crawl time. Note that our introduction of time does not mean that we are working with streaming graphs like in [9]: even if evolving, the graph is considered as effectively static – because it is large enough that changes during the crawl are unlikely to modify its general characteristics. "Time" in our case is a measure of cost, not a dimension of the study.

A. Edge Costs

Collecting the list of edges incident to a specific node on a social media network is usually an operation allowed explicitly by the platform's APIs. The number of calls to the platform's API needed to obtain the data depends on the network structure of the social media platform. The following elements contribute to the cost of retrieving the edge list for a single node:

- Directed or Undirected nature of the relation between users.
- Pagination policy of the returned result.

Usually, the list of connections of a given node is returned through a single call to the API. When relevant (e.g. in the case of Twitter, Flickr, Instagram, etc.) APIs make a distinction between users *following* the node and users *followed* by the node. These two lists are usually returned through different calls to the platform.

Every platform that we have surveyed has a paging policy in place. This means that, even if the connections of a single node are potentially available through a single call, the actual number of calls needed is defined by the number of connections of the node and by the paging policy of the platform. In the case of a node v member of a directed social media the number of calls required to collect the full list of its connections will be:

$$calls[1, Deg_{(v)}] = \left\lceil \frac{Deg_{-v}}{pag} \right\rceil + \left\lceil \frac{Deg_{+v}}{pag} \right\rceil,$$

where Deg_{-v} and Deg_{+v} are the in- and out-degree of v, respectively. pag is the number of edges returned with each call as defined in the platform specific pagination policy. Later in the text we will refer to this value as page size.

OSN	Page Size	Secs per Query	Avg Throughput	Class	Ref
Tumblr	20	17.28	1.15	SP-HQI	[4]
Youtube	50	4.32	11.57		[6]
Twitter	5,000	60	83.33	LP-HQI	[5]
Google+	2,000	8.64	231.4		[2]
Lastfm	50	.2	250	SP-LQI	[3]
Flickr	1,000	1	1,000	LP-LQI	[1]

TABLE I: The API policy details of the six selected social media. The average throughput is calculated in number of edges per second.

B. Cost Multipliers

Once we have defined how many calls we need to collect the list of edges of a given node, our model needs to account for the fact that, to the best of our knowledge, all social media platform's APIs have rate limits. The differences in rate limits between different media is almost always quantitative. API rate limits regulate the number of calls per unit of time. As summarized in Table I, these rate policies can be extremely different between different platforms and they can even change over time when a single platform is considered. Social Media platforms use more or less strict limits for a number of reasons. Platforms desire to keep under control the costs associated with running the API, they need to prevent abusive behavior from malicious users, and to prevent access to the whole network.

By restricting how many calls to the API are possible per second, rate limits define the actual budget, expressed in time, needed to collect the list of edges incident to a node. Using s as the number of seconds we need to wait before submitting another query to the API system, the final cost of crawling node v is:

$$cost[v] = (s + l) \left(\left\lceil \frac{Deg_{-v}}{pag} \right\rceil + \left\lceil \frac{Deg_{+v}}{pag} \right\rceil \right).$$

Here we introduce l as the communication's latency: the number of seconds it takes to complete one call. Latency on the data collection is extremely variable, and considering it falls outside the scope of this article. Since it is a fair assumption that, in most real systems, $l \ll s$, in the following experiments we will assume for latency to be negligible and stable ($l = 0$).

Note that the *budget-per-node* – $cost[v]$ – has no trivial relation with the actual cost of collecting the entirety of the network data from social media platforms. We should rather consider $cost[v]$ together with the pagination policy and the topology of the network. To explain this, let us consider two hypothetical APIs regulated through two different policies. The first policy P_1 has a rate limit of 1 second ($s = 1$)

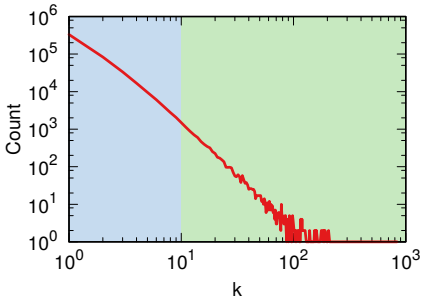


Fig. 2: A power-law degree distribution: the number of nodes (y-axis) with a given degree (x-axis). We highlight in blue the part of the distribution where policy P_1 has a lower cost per node than policy P_2 – because of P_1's more favorable rate limit –, in green where policy P_2 has a lower cost per node than policy P_1.

and returns pages of 10 edges each ($pag = 10$). We can consider the throughput of P_1 as 10 edges per second. The second policy, P_2, allows a call every 2 seconds ($s = 2$) but it returns 100 edges per page ($pag = 100$). The second policy has a throughput of 50 edges per second, five times larger that the first. A hasty evaluation could suggest that P_2 is always preferable to P_1.

Figure 2 compares the actual cost of the two policies when deployed on a network characterized by a power-law degree distribution and containing 500k nodes. The light-blue area represents the part of the distribution in which P_1 pays less than P_2 (P_1 needs less time than P_2 to collect this part of the network). Given the power-law distribution of this network, P_1 results a more economic strategy than P_2 to collect 492k of the 500k total nodes. As a consequence of this, to collect the entire network (including the green area of Figure 2) P_1 results to be two times faster than P_2.

While the results seem to suggest that, in several cases, small pages and frequent calls might be preferable to large pagination policies, it should be considered that latency in the process can largely affect the final result. Looking at $cost[v]$'s formula, if we were to significantly reduce s, we would break the $l \ll s$ assumption, with negative repercussions on crawl time. We leave the investigation of these consequences as future work.

IV. Algorithms (f)

In this section we briefly present the algorithms (f, see Section III) that we use for our benchmark B. As mentioned in Section II, we only consider topological sampling. Node- and edge-induced are not appropriate for social media API crawl: APIs do not usually return one edge at a time, nor we can obtain arbitrary node IDs without exploration as node-induced methods would demand. Following Figure 1's classification, we divide topological network sampling in deterministic (Section IV-A) and probabilistic (Section IV-B).

A. Deterministic

Deterministic topological sampling is the oldest class of network sampling methods. It includes the classical approaches to graph navigation: Breadth-, Depth-First Search, and Snowball sampling.

In Breadth-First Search (BFS) [32], [31], we start from a seed v and we put its neighbors in a First-In, First-Out (FIFO) queue. In this case we call v a parent, and its neighbors are its children. Children of the same parent are siblings. We then explore the neighbors one by one, inserting the neighbor's neighbors in the FIFO queue. All the siblings of a node will be explored before switching to their children. Figure 3(a) shows an example of this approach.

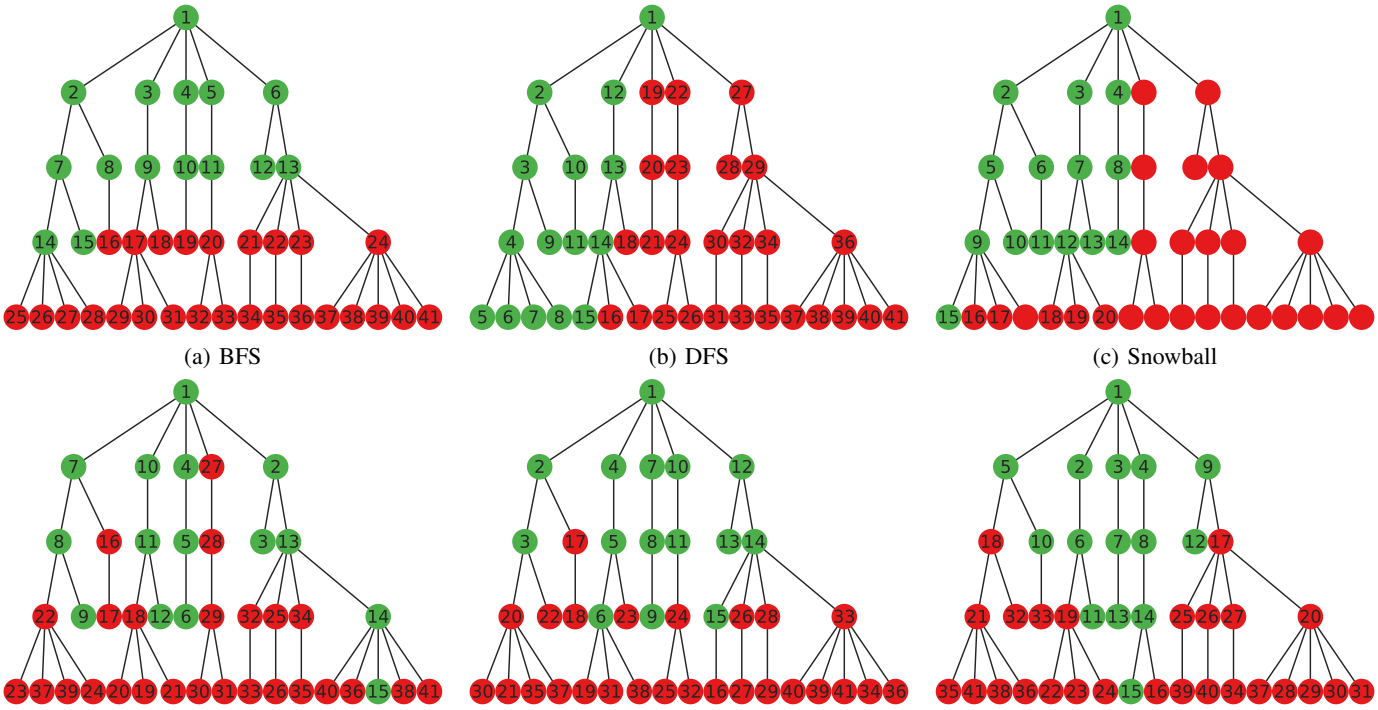
Depth-First Search (DFS) [46] is similar to BFS, but uses a Last-In, First-Out (LIFO) queue instead of a FIFO: the children of a node are explored before its siblings. In practice, DFS explores a chain from v until the farthest node it can reach and, when there are no more possible children to explore, it backtracks and explores the sibling of the last parent selected, as Figure 3(b) shows.

Snowball sampling [22] is almost identical to BFS, in fact sometimes these terms are used as synonyms. However, in classical Snowball sampling, the recruited subjects are asked to provide n contacts. Therefore, if the node has degree $k > n$, Snowball does not explore all of its neighbors. In social media data gathering, there is often little cost in exploring all neighbors of a node: in that case Snowball is equivalent to BFS. However, we consider it as a separate method, because then we can simulate a scenario in which the researcher can decide to request only a given number of pages to the API, effectively capping the maximum degree of the sample. This might have positive repercussions: for instance, it will limit degree bias by not exploring all of a hub's neighbors. Fixing the degree bias is one of the main objective of other methods, as we will see. Figure 3(c) shows an example of this approach.

B. Probabilistic

Following our classification in Figure 1, we divide the probabilistic network sampling algorithms in two subclasses: random-walk-based (Section IV-B1) and non-random-walk-based (Section IV-B2).

1) Random Walk Based: When sampling a network using random walks (RW), the procedure starts from v and chooses one of its neighbors at random [28], [27]. The same process is repeated for every node visited. In vanilla random walk, one is allowed to re-visit previously visited nodes although, for our benchmark system B, if a node was already visited we do not count its cost twice. In many practical applications, random walk sampling takes a parameter p which establishes a probability of restarting the walk from the seed v, or teleporting to an arbitrary node. In our paper we use the teleportation flavor, for efficiency reasons. For high budgets most of the nodes are already explored, and thus the probability of finding a non-explored node is low. Since visiting an already-explored node does not incur in any cost, the expected time to spend the full budget would skyrocket. Additionally, we put a cap on failed attempts: if the walker visits 1,000 already-explored nodes in a row, we consider it trapped and abort the exploration even if there is budget left. Figure 3(d) shows an example of this approach.



(a) BFS (b) DFS (c) Snowball

(d) Random Walk (e) MHRW (f) Forest Fire

Fig. 3: Examples of how different network sampling strategies explore a given network. Each node is labeled with the order in which it is explored. The node color shows whether the node was sampled (green) or not (red), assuming a budget of 15 units and a constant cost of 1 unit per node. Snowball assumes $n = 3$ (unlabeled nodes are not explored due to this parametric restriction), while Forest Fire has a burn probability of .5.

Exploring a network through random walks has known biases. The probability of visiting a node is proportional to its degree. As a consequence, high degree nodes are oversampled, leading to representativeness issues. Network scientists have developed a Metropolis-Hastings correction for random walk sampling (MHRW) [45], [24]. The network is explored via random walks as before, choosing a neighbor at random from the currently visited node v, say u. However, u is not accepted with probability 1. Instead, it is accepted with probability k_v/k_u, where k_v is v's number of connections (degree). In practice, if u has a higher degree than v, there is a chance that we will attempt to select a different u' to continue the random walk. Figure 3(e) shows an example of this approach.

The final approach we consider in the random walk family is the re-weighted correction [41], [40] (RWRW, also known as Respondent Driven Sampling, or RDS). In RWRW, the network exploration is conducted with the vanilla RW approach described above. Once the exploration is done, each statistical property of G – π – is reconstructed with the formula:

$$\pi_i = \frac{\sum_{v \in A_i} k_v^{-1}}{\sum_{v \in V} k_v^{-1}},$$

where i is the value of the property, e.g. the degree, A_i is the set of nodes with exactly i value for the property, V is the full set of nodes, $k_v = i$ in the numerator and equal to the value of the property of node v in the denominator, and p_i is the estimated probability of observing the value i in the full network. This works for discrete variables. For continuous ones, one could perform a kernel density estimator, but in this paper we evaluate RWRW only on discrete properties. There is no need to show the G' sample extracted by RWRW, because it is equivalent to the one extracted by RW. Therefore, the downside of RWRW is that it can only return a correct estimation of G's properties, but not a G' itself that can be then used as input for any arbitrary algorithm.

2) Non Random Walk Based: In this subclass we have sampling strategies which have a probabilistic component, but which do not base their core strategy on a random walk. The most famous approach is Forest Fire [27] (FF). In FF, one starts by performing a BFS exploration of the graph. However, FF introduces a parameter: the "burning probability". If a node passes the burning probability test, the node is "burned" and the BFS exploration will move on to its neighbors. Figure 3(f) shows an example of this approach.

Finally, we have Neighbor Reservoir Sampling [29], [9] (NRS). NRS starts by building a small core of explored nodes using RW. However, the majority of NRS's budget is spent on a second phase. Suppose that in the first phase the algorithm has sampled $|V'|$ nodes. We select at random a neighbor u directly connected to any of the V' sampled nodes. We also select at random a sampled node v from V'. We add u to V' if and only if: (i) by adding u and removing v, G' still has only a single connected component, and (ii) we extract a uniform random number $\alpha < |V'|/i$, where i is initially set to $|V'|$ and increased by one for every attempt to add a new neighbor u. In practice, $|V'|$ stays constant, but each subsequent attempt

to add new us becomes less likely to succeed – because of increasing i. NRS is by far the most computationally expensive method because at each potential node addition it has to verify that G' still has a single connected component. Also, given the diminishing acceptance probability, the last units of budget take a long time before they are spent. For these reasons, we are unable to perform enough experiments to have statistically robust results, and so we drop it from Section VI.

V. Network Topologies (G)

To test the sampling methods, we use exclusively synthetic networks. We focus on models that have realistic topological assumptions, thus disqualifying simpler generative processes such as Erdős-Rényi or Caveman. The three approaches we use are: scale free via a Barabási-Albert preferential attachment (BA) [11], Watts-Strogatz Small World (SW) [47], and the Lancichinetti-Fortunato-Radicchi (LFR) bechnmark originally defined for community discovery [25].

We chose the BA model because it generates realistically broad degree distributions and small diameters, although low clustering and no communities. The SW model, unlike BA, has high clustering, but not realistic degree distributions. LFR is our main benchmark, because it faithfully reproduces most characteristics of real world networks: power law degree distribution, high clustering, and network communities.

Because most synthetic network models generate undirected networks, we modify each method to allow for edge directionality. We perform experiments on both the directed and the original undirected versions.

For the scale free network, although there are directed formulations [15], we found them unsatisfactory because they generate graphs with very low reciprocity. Networks should accurately reflect the common social media scenario where most friendships are reciprocated [31]. So, we use a four step generative process. First, we generate a Pareto distribution. In the second step, we use it to assign to each node its outdegree – by picking endpoints at random –, and in the third step we use it to assign the indegree. After the second step we have a power-law outdegree distribution and a normal in-degree distribution. After the third step we have pseudo-power-law distributions for both in- and out- degree. The fourth and final step plants reciprocity: each edge has a 50% chance of being reciprocated.

For the SW network, we take the result of the vanilla SW model. Then, with a 50% probability we reverse the edge direction. With a further 50% probability, we keep the original edge, enabling reciprocity. For LFR, the authors created a directed version [25].

All networks, directed and undirected, have $\sim$200k nodes. The number of edges is variable, due to the differences in how the network topologies are generated by the various methods. Thus we go from the smallest network of 1.76M edges (undirected LFR) to the largest of 4.22M edges (directed LFR).

VI. Results

In this section we perform our experiments. They are all conducted on six distinct synthetic topologies, as explained in Section V. We are interested in checking the seven methods (BFS, SBS, FF, DFS, RW, RWRW, MHRW) at ten different budgets, for ten different tests, and for six different API systems, based on the restrictions imposed by the following social media: Tumblr, Youtube, Twitter, Google+, Lastfm, and Flickr. Section VI-A details the experimental setup describing the tests and the initialization of the API policies.

Note that most of the sampling methods have some degree of randomness. Thus, we run each test 100 times, and we take the average performance as its result. This generates small enough standard errors to make the distinctions between the performances statistically significant, except where otherwise noted.

In each of the following sections, we discuss results by a different scenario dimension: by API System in Section VI-B, by network topology in Section VI-C, by evaluation test in Section VI-D, and by budget level in Section VI-E. Each section then provides a few illustrative cases, using various combinations of values on the other dimensions. This is done to manage the size of the result section, given that to show all of our results we would need 180 plots.

A. Experimental Setup

To setup our experiments we have to build the benchmark system B by specifying: the methods we test (fs), the synthetic network topologies (Gs), our budget choices (bs), the q_G tests we are going to run, and the details of the API systems we test (Ps). We have presented the first two dimensions of our study in Sections IV and V, respectively. Here, we fill the remaining gaps.

1) Budget (b): As discussed in Section III, our fundamental unit of cost is the number of seconds it is required to crawl a network. It makes sense to use seconds as our budget unit here. To keep the experiments' complexity manageable in terms of time and space, in Section V we made the decision of limiting the size of the synthetic networks to something significantly smaller than the actual structures underlying most popular social media. As a consequence, we also scale our budget to be more restrictive than usual when it comes to web crawling. The idea is that the relationship between budget and underlying network size is linear. If we increase the size of the network tenfold, increasing the budget tenfold will yield comparable results.

Our budget dimension then goes from one hour (3,600 budget units – seconds), to three days (259,200 budget units). Since for the loosest API policies we can completely explore one node per query, increasing the budget to be much larger than the number of nodes in the network ($\sim$200k) is not going to yield any interesting insight: we would have crawled the full network well before we hit the budget limit.

2) API Systems (P): As previously noted, we focus on six social media and their API policies. Table I reports the details of the pagination size and length of the mandatory interval between two queries – in seconds. We also report the URL that is the source of our estimation, and the estimated throughput of the system. This estimation ignores network latency, which we assume to be equal to zero for all social media. The API throughput is measured in edges per second.

For Section VI-B we classify API systems in four classes along two dimensions: Large and Small Pages (LP and SP), and High and Low Query Interval time (HQI and LQI). We then pick one API policy from Table I to represent each quadrant as follows:

- LP-HQI Twitter: it has the largest page size and the longest interval between queries;
- SP-HQI Tumblr: smallest page size and second-highest interval between queries;
- LP-LQI Flickr: third-largest page size, second-lowest query interval size;
- SP-LQI Lastfm: second-smallest page size and lowest overall query interval time.

This avoids the confusion that could come from averaging results: one could think that method x is the best for, say, Flickr, even if the results are averaged over topologies that have nothing to do with Flickr, e.g. undirected SW.

3) Tests (q_G): When analyzing the structure of online social media, there are commonly three basic measures most studies are interested on: the degree distribution, identifying the most important actors in the social network, and the relationship between node attributes and the topology of the network. These three measures are at the basis of our ten tests.

For the degree distribution, we use the standard approach of calculating the Kolmogorov-Smirnov distance between the original degree distribution and the one we obtain from the network sample. Given that we have both directed and undirected networks, this generates three tests: undirected KS distance (k KS), in-degree and out-degree KS distances (k-IN KS and k-OUT KS, respectively).

For the centrality test, we calculate the Spearman rank correlation between the degree vector of the original network and the one of the sample. By using the Spearman correlation we correct for the broad distributions typical of real world networks, and we make sure we are testing what is at interest here: that the top-ranking nodes are correctly sorted out from the bottom ranking ones. Note that we use the degree for time complexity reasons, however this could be considered fairly similar to measures of centrality such as PageRank, given their high correlation [19]. This generates one test for the directed and one test for the undirected networks.

For the relationship between node attributes and topology, we run two tests each for directed and undirected networks. We generate two different attributes: one characterized by homophily and a disassortative one. Homophily means that nodes at the endpoints of the same edge tend to have the same value of the attribute [36]. Disassortativity holds the opposite. For the test, we simply calculate the assortativity of the attributes as defined in [36], and we compute the Mean Absolute Error (MAE) of the sample when compared to the original network. This is a necessary test, because homophily is one of the defining characteristics of social networks [12], [30]: misjudging it might lead to wildly incorrect conclusions.

Finally, the directed networks have an additional test. Most social systems are characterized by a certain degree of reciprocity [31]: a connection from a to b is usually reciprocated by a connection from b to a. We also calculate the reciprocity MAE of the sample.

Method	SP-HQI	LP-HQI	SP-LQI	LP-LQI
BFS	-0.7226	-0.7669	1.5333	1.0887
SBS	**-0.4026**	-0.8048	1.4970	1.0733
FF	-0.6440	-0.9081	**1.5445**	**1.0935**
DFS	-1.3188	-0.8447	-0.1603	0.8866
RW	-1.0136	-0.6439	0.8316	0.4687
RWRW	-0.9715	**-0.6149**	0.7501	0.3827
MHRW	-1.1536	-0.9745	0.4343	0.1910

TABLE II: Average standardized results for different API policies. Average across all directed and undirected networks. Higher is better.

Note that in Sections VI-B, VI-C, and VI-E we are collapsing over the test dimension to calculate average performance, which means that we are averaging test performance values. However, the KS test, Spearman correlation and MAE have different domains with different semantics. A value of .1 in each of these three tests means something different. Thus we need to manipulate them to perform a fair comparison.

First, we multiply KS and MAE by -1, aligning them with the Spearman semantics that "higher is better". Then, each measure is standardized, by removing its average and dividing by its standard deviation. The result is a unitless measure that is harder to interpret – it reports how many standard deviations above or below the average measurement the test is – but that is otherwise identical in meaning with all the other tests. At this point, different tests can be averaged together.

B. Results by API System

In this section we discuss how each method performs when tested against each API system. The API policy legend is the classification introduced in Section VI-A2, and the throughput information is in Table I. Table II reports the average performances across all topologies, tests and budgets.

The Table highlights the first dimension of our no-free-lunch situation: for different API policies, different methods performed best. In particular, Snowball sampling works best for the low throughput SP-HQI class, and Forest Fire/BFS – the difference between them is not significant – for high-throughput ones (SP-LQI, LP-LQI). In LP-HQI, RWRW performs best.

Note that here we also confirm the page size effect discussed in Section III-A: even if LP-LQI has $4\times$ throughput over SP-LQI (see Table I), each method performs much better on the latter. The difference is even more remarkable when comparing LP-HQI to SP-HQI: in that case, the throughput difference is $> 80\times$ in favor of LP-HQI (see Table I), yet – in some cases – the sampling methods perform better on SP-HQI.

Figure 4 shows two examples of the same test ran on the same topology, but for different API policies: SP-HQI (Tumblr) and SP-LQI (Lastfm). We can see that in the SP-HQI case the two best performing methods are MHRW and DFS. However, in SP-LQI, DFS is the worst performing one, and MHRW is not the best one, beaten by Snowball sampling for low budgets, and by everything else for high budgets.

C. Results by Topology

In this section we discuss how each method performs when tested against each topology: the directed and undirected

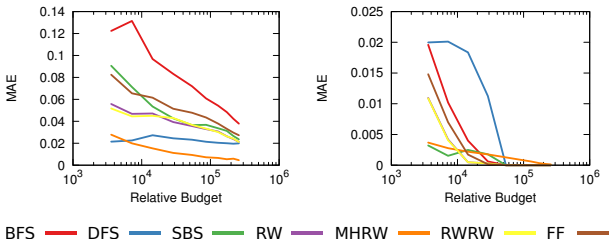


Fig. 4: Comparison of sampling method MAE for the disassortative node attribute in the undirected BA network, for SP-HQI (left) and SP-LQI (right). Lower is better.

Method	LFR	BA	SW	dLFR	dBA	dSW
BFS	-0.5966	-0.0564	**0.3075**	0.3120	0.0542	0.9008
SBS	-0.2466	**0.0859**	0.2208	0.3001	0.0544	**0.9070**
FF	-0.7707	0.0271	0.2746	**0.4066**	**0.0603**	0.8745
DFS	-0.3479	-0.3616	0.1500	-0.0834	0.0515	0.2440
RW	**-0.2381**	0.0789	-0.1298	0.2856	-0.7329	0.1590
RWRW	-0.4335	-0.0120	-0.3229	0.3354	-0.5844	0.0028
MHRW	-2.2700	-0.1096	-0.2776	-0.0820	-0.1674	0.0542

TABLE III: Average standardized results for different network topologies. Average across all directed and undirected networks. Higher is better.

versions of the LFR, BA, and SW models – as introduced in Section V. Table III reports the average performances across all API policies, tests and budgets.

Just like in the previous section, here we have another dimension in which there is no free lunch. In the directed networks, BFS-based methods like Forest Fire and Snowball sampling work well, but – in the undirected networks – vanilla RW works best for the most realistic network model (LFR), and fairly well also for the scale free network.

Figure 5 shows an example of how different topologies can radically influence the performances of different sampling methods. In the Small World topology, the highest errors are registered for BFS, Snowball and Forest Fire. However, these are among the best performing approaches in a preferential attachment network, with RW and RWRW performing worst. Again, these two methods work fairly well in a directed LFR topology, with MHRW now showing as outlier among the worst performing approaches.

D. Results by Test

In this section we focus on each single test we run, making a distinction between directed and undirected versions of the same test. Here we can directly interpret the returned values for each test, since we do not need to standardize across tests with different domains for their results.

Table IV reports the results for the directed networks. While BFS and similar approaches perform reasonably well in many scenarios, they are beaten in the out-degree distribution test by RWRW and MHRW, which are methods explicitly designed for this particular test in mind. The BFS-based approaches also have trouble with assortative node attributes, likely because they are trapped in areas of the network where all nodes have the same attribute value.

We can draw similar conclusions from Table V. This implies that the no-free-lunch scenario applies also to the various tests: methods designed to perform well with a specific target in mind do so, but they underperform when one shifts her attention to other topological features of the network.

Figure 6 shows one of the many examples one can draw. DFS is the best performing algorithm when testing for centrality correlations, but it is also one of the methods with higher MAE when testing for reciprocity in the same network (directed LFR) and same API policy (Google+). On the other hand, Snowball sampling has low centrality correlations in the same scenario, but also a low reciprocity MAE – among the best of the lot.

E. Results by Budget

Finally, we look at what happens when one has different levels of budget available. Superficially, by looking at Table VI, one could think that here there is a clear winner: no matter the budget level, Snowball sampling performs the best. However, such conclusion would be hasty. As we saw in the previous sections, Snowball performs decently on average, but it is often beaten in many scenarios. Most of its power in this test is driven by its excellent performance in the MAE test for disassortative attributes, in which it performs twice better than all other methods (see Table V).

The issue with different budget levels becomes even more problematic when looking at specific instances of this test. Consider Figure 7. In both these tests we have clear winners at low budgets and clear winners at high budgets – ironically, Snowball sampling is never the winner in any of the shown cases. However, the best low-budget performers are also the worst high-budget performers, and vice versa. Our conclusion is that, even if at a theoretical level Snowball sampling is better *on average* than everything else at all budget levels, in *practical and specific* scenarios the available budget heavily influences the sampling method choice.

VII. Conclusion

In this paper, we explore the different performances of different network sampling methods over a variety of scenarios. Real world network sampling has many practical problems to consider: how does the target API system work? How much budget do I have available? On which analysis do I want to focus? Classical literature on the problem only superficially considers some of these questions. Here, we show that answering them is crucial: different methods perform differently when targeting specific API platforms, with specific budgets, and for specific tests.

For instance, one cannot consider an API system as a simple pipe with a given throughput. Its features have a complex relationship with the performance of the sampling methods: small pages are usually better than large ones, even if they cause a theoretically smaller throughput – all this assuming a negligible overhead from network latency.

There are several future works that could originate from this paper. First, as just mentioned, here we largely ignore the influence that network latency has on the sampling procedure.

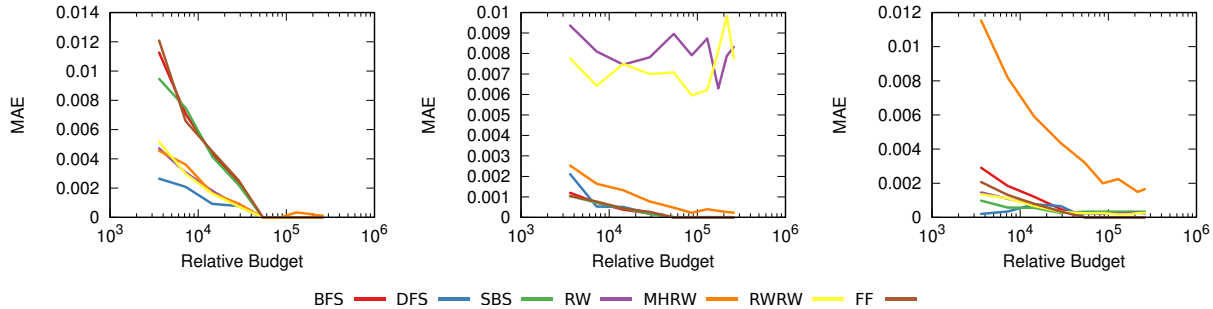


Fig. 5: Comparison of sampling method MAE for the assortative node attribute in the directed Lastfm network, for different topologies (from left to right): Small World, Barabasi-Albert, LFR benchmark. Lower is better.

Method	k-IN KS	k-OUT KS	Centr Corr	Assort MAE	Disassort MAE	Recipr MAE
BFS	**0.4876**	0.1404	**0.5618**	0.0213	**0.0035**	0.1863
SBS	0.4960	0.1338	0.5556	0.0208	0.0036	0.1873
FF	0.4881	0.1179	0.5616	0.0213	0.0041	**0.1852**
DFS	0.6509	0.2713	0.4467	**0.0068**	0.0038	0.2594
RW	0.7860	0.1675	0.2876	0.0076	0.0053	0.3226
RWRW	0.8540	**0.1179**	0.2881	0.0073	0.0053	0.3225
MHRW	0.8025	0.1220	0.2883	0.0104	0.0081	0.3036

TABLE IV: Average standardized results for different quality tests. Average across directed networks only. Lower is better, except for the "Centr Corr" test.

Method	k KS	Centr Corr	Assort MAE	Disassort MAE
BFS	**0.4700**	**0.4954**	0.0452	0.0468
SBS	0.5140	0.4591	0.0331	**0.0285**
FF	0.4779	0.4805	0.0496	0.0432
DFS	0.4755	0.3516	0.0330	0.0458
RW	0.5926	0.4592	0.0268	0.0408
RWRW	0.7379	0.4590	**0.0268**	0.0404
MHRW	0.7856	0.1418	0.0342	0.0456

TABLE V: Average standardized results for different network topologies. Average across undirected networks only. Lower is better, except for the "Centr Corr" test.

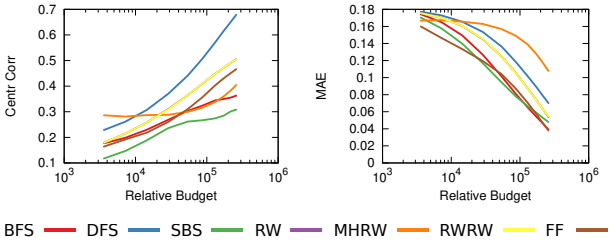


Fig. 6: Comparison of sampling method perfomances for different tests in the Google+ network with directed LFR topology. (Left) Centrality correlation (higher is better), (Right) MAE of reciprocity (lower is better).

While small pages increase the power of the sample methods, at some point making too many requests per unit of time will hit decreasing returns. Finding where this optimal point lies is a quest for follow-up experiments.

Second, here we had to ignore some sampling methods for various reasons: induced samples because they do not play well with traditional API systems, and NRS because of its high time complexity. Adding these methods – and more tests for a well-rounded analysis of less common social media analysis tasks, or the scenario of streaming graphs – would provide

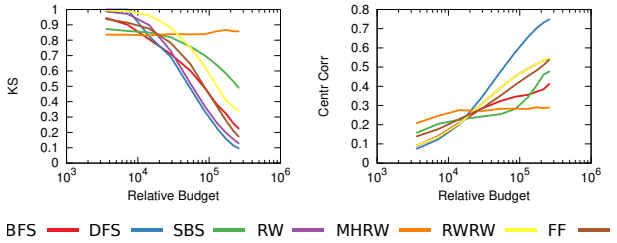


Fig. 7: Comparison of sampling perfomances at different budget levels. (Left) KS distance for undirected LFR network in Youtube (lower is better). (Right) Centrality correlation for directed LFR network in Youtube (higher is better).

further investigation pathways.

Finally, the benchmark system can be used as a target for the development of new network sampling algorithms. Now that we have a more realistic target, we should be able to create methods that behave better in more complex scenarios, by targeting directly API costs. Our proposal would be to estimate the amount of new information sampling a node could bring, using a Bayesian framework similar to the one developed for network backboning [17].

Acknowledgements. The authors wish to thank Clara Vandeweerdt for insightful comments.

REFERENCES

[1] "Flickr api," https://www.flickr.com/services/api/flickr.contacts.getPublicList.html, accessed: 2018-03-12.
[2] "Google+ api," https://developers.google.com/+/web/api/rest/latest/people/list, accessed: 2018-03-12.
[3] "Lastfm api," https://www.last.fm/api/show/user.getFriends, accessed: 2018-03-12.
[4] "Tumblr api," https://www.tumblr.com/docs/en/api/v2, accessed: 2018-03-12.

Method	3.6k	7.2k	14.4k	28.8k	54k	86.4k	129.6k	172.8k	216k	259.2k
BFS	-1.2499	-0.8217	-0.3185	0.1166	0.4803	0.6965	0.9060	1.0479	1.1647	1.2198
SBS	**-1.1648**	**-0.6278**	**-0.1666**	**0.2598**	**0.6108**	**0.7894**	**0.9536**	**1.0872**	**1.1909**	**1.2434**
FF	-1.2617	-0.8278	-0.3507	0.1146	0.4995	0.7060	0.9012	1.0534	1.1696	1.2351
DFS	-1.5304	-1.1119	-0.8046	-0.3803	-0.0163	0.2218	0.4630	0.6744	0.8345	0.8974
RW	-1.2670	-0.9372	-0.6447	-0.3844	-0.1085	0.0447	0.1934	0.3143	0.4295	0.4689
RWRW	-1.2120	-0.9056	-0.6661	-0.4151	-0.1446	-0.0295	0.1229	0.2401	0.3315	0.3606
MHRW	-1.5643	-1.3279	-1.0551	-0.7807	-0.4927	-0.3034	-0.2080	-0.0673	0.0686	0.0358

TABLE VI: Standardized results for different budgets. Average across all directed and undirected networks. Higher is better.

[5] "Twitter api," https://developer.twitter.com/en/docs/basics/rate-limits.html, accessed: 2018-03-12.

[6] "Youtube api," https://developers.google.com/youtube/v3/docs/channels/list, accessed: 2018-03-12.

[7] N. Ahmed, J. Neville, and R. R. Kompella, "Network sampling via edge-based node selection with graph induction," 2011.

[8] N. K. Ahmed, J. Neville, and R. Kompella, "Space-efficient sampling from social activity streams," in *Proceedings of the 1st international workshop on big data, streams and heterogeneous source mining: algorithms, systems, programming models and applications*. ACM, 2012, pp. 53–60.

[9] ——, "Network sampling: From static to streaming graphs," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 8, no. 2, p. 7, 2014.

[10] L. Backstrom and J. Kleinberg, "Network bucket testing," in *Proceedings of the 20th international conference on World wide web*. ACM, 2011, pp. 615–624.

[11] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *science*, vol. 286, no. 5439, pp. 509–512, 1999.

[12] M. Barone and M. Coscia, "Birds of a feather scam together: Trustworthiness homophily in a business network," *Social Networks*, vol. 54, pp. 228–237, 2018.

[13] D. C. Bell, E. B. Erbaugh, T. Serrano, C. A. Dayton-Shotts, and I. D. Montoya, "A comparison of network sampling designs for a hidden population of drug users: Random walk vs. respondent-driven sampling," *Social science research*, vol. 62, pp. 350–361, 2017.

[14] N. Blagus, L. Šubelj, and M. Bajec, "Empirical comparison of network sampling techniques," *arXiv preprint arXiv:1506.02449*, 2015.

[15] B. Bollobás, C. Borgs, J. Chayes, and O. Riordan, "Directed scale-free graphs," in *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 2003, pp. 132–139.

[16] F. Ciulla, N. Perra, A. Baronchelli, and A. Vespignani, "Damage detection via shortest-path network sampling," *Physical review E*, vol. 89, no. 5, p. 052816, 2014.

[17] M. Coscia and F. M. Neffke, "Network backboning with noisy data," in *Data Engineering (ICDE), 2017 IEEE 33rd International Conference on*. IEEE, 2017, pp. 425–436.

[18] A. Dasgupta, R. Kumar, and D. Sivakumar, "Social sampling," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2012, pp. 235–243.

[19] G. Ghoshal and A.-L. Barabási, "Ranking stability and super-stable nodes in complex networks," *Nature communications*, vol. 2, p. 394, 2011.

[20] M. Gjoka, M. Kurant, C. T. Butts, and A. Markopoulou, "Walking in facebook: A case study of unbiased sampling of osns," in *Infocom, 2010 Proceedings IEEE*. Ieee, 2010, pp. 1–9.

[21] ——, "Practical recommendations on crawling online social networks," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 9, pp. 1872–1892, 2011.

[22] L. A. Goodman, "Snowball sampling," *The annals of mathematical statistics*, pp. 148–170, 1961.

[23] D. D. Heckathorn and C. J. Cameron, "Network sampling: From snowball and multiplicity to respondent-driven sampling," *Annual review of sociology*, vol. 43, pp. 101–119, 2017.

[24] B. Krishnamurthy, P. Gill, and M. Arlitt, "A few chirps about twitter," in *Proceedings of the first workshop on Online social networks*. ACM, 2008, pp. 19–24.

[25] A. Lancichinetti and S. Fortunato, "Community detection algorithms: a comparative analysis," *Physical review E*, vol. 80, no. 5, p. 056117, 2009.

[27] J. Leskovec and C. Faloutsos, "Sampling from large graphs," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2006, pp. 631–636.

[26] S. H. Lee, P.-J. Kim, and H. Jeong, "Statistical properties of sampled networks," *Physical Review E*, vol. 73, no. 1, p. 016102, 2006.

[28] L. Lovász, "Random walks on graphs," *Combinatorics, Paul erdos is eighty*, vol. 2, no. 1-46, p. 4, 1993.

[29] X. Lu and S. Bressan, "Sampling connected induced subgraphs uniformly at random," in *International Conference on Scientific and Statistical Database Management*. Springer, 2012, pp. 195–212.

[30] M. McPherson, L. Smith-Lovin, and J. M. Cook, "Birds of a feather: Homophily in social networks," *Annual review of sociology*, vol. 27, no. 1, pp. 415–444, 2001.

[31] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee, "Measurement and analysis of online social networks," in *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*. ACM, 2007, pp. 29–42.

[32] E. F. Moore, "The shortest path through a maze," in *Proc. Int. Symp. Switching Theory, 1959*, 1959, pp. 285–292.

[33] J. L. Moreno, H. H. Jennings *et al.*, "Who shall survive?" 1934.

[34] F. Morstatter, J. Pfeffer, and H. Liu, "When is it biased?: assessing the representativeness of twitter's streaming api," in *Proceedings of the 23rd International Conference on World Wide Web*. ACM, 2014, pp. 555–556.

[35] F. Morstatter, J. Pfeffer, H. Liu, and K. M. Carley, "Is the sample good enough? comparing data from twitter's streaming api with twitter's firehose." in *ICWSM*, 2013.

[36] M. E. Newman, "Assortative mixing in networks," *Physical review letters*, vol. 89, no. 20, p. 208701, 2002.

[37] L. Peel, D. B. Larremore, and A. Clauset, "The ground truth about metadata and community detection in networks," *Science advances*, vol. 3, no. 5, p. e1602548, 2017.

[38] D. Pennacchioli, G. Rossetti, L. Pappalardo, D. Pedreschi, F. Giannotti, and M. Coscia, "The three dimensions of social prominence," in *International Conference on Social Informatics*. Springer, 2013, pp. 319–332.

[39] A. H. Rasti, M. Torkjazi, R. Rejaie, D. Stutzbach, N. Duffield, and W. Willinger, "Evaluating sampling techniques for large dynamic graphs," *Univ. Oregon, Tech. Rep. CIS-TR-08*, vol. 1, 2008.

[40] A. H. Rasti, M. Torkjazi, R. Rejaie, N. Duffield, W. Willinger, and D. Stutzbach, "Respondent-driven sampling for characterizing unstructured overlays," in *INFOCOM 2009, IEEE*. IEEE, 2009, pp. 2701–2705.

[41] M. J. Salganik and D. D. Heckathorn, "Sampling and estimation in hidden populations using respondent-driven sampling," *Sociological methodology*, vol. 34, no. 1, pp. 193–240, 2004.

[42] J. A. Smith, J. Moody, and J. H. Morgan, "Network sampling coverage ii: the effect of non-random missing data on network measurement," *Social networks*, vol. 48, pp. 78–99, 2017.

[43] S. Soundarajan, T. Eliassi-Rad, B. Gallagher, and A. Pinar, "Maxreach: Reducing network incompleteness through node probes," in *Advances in Social Networks Analysis and Mining (ASONAM), 2016 IEEE/ACM International Conference on*. IEEE, 2016, pp. 152–157.

[44] ——, "ε-wgx: Adaptive edge probing for enhancing incomplete networks," in *Proceedings of the 2017 ACM on Web Science Conference*. ACM, 2017, pp. 161–170.

[45] D. Stutzbach, R. Rejaie, N. Duffield, S. Sen, and W. Willinger, "On unbiased sampling for unstructured peer-to-peer networks," *IEEE/ACM Transactions on Networking (TON)*, vol. 17, no. 2, pp. 377–390, 2009.

[46] R. Tarjan, "Depth-first search and linear graph algorithms," *SIAM journal on computing*, vol. 1, no. 2, pp. 146–160, 1972.

[47] D. J. Watts and S. H. Strogatz, "Collective dynamics of small-worldnetworks," *nature*, vol. 393, no. 6684, p. 440, 1998.

[48] Z. Wei, J. Zhao, K. Liu, Z. Qi, Z. Sun, and G. Tian, "Large-scale knowledge base completion: Inferring via grounding network sampling over selected instances," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, 2015, pp. 1331–1340.